VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 Jan Pecník

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Pecník**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: TRANSEXPRESS Intl. spol. s r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Konzultant bakalářské práce: Ing. Jaromír Sitek

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * 1D16 * OSTRAVA *

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. Dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 20. Dubna 2018 .....

TRANSEXPRESS
Intl. spol. s r.o.
Orlovská 791/40, 713 00 Ostrava, Heřmanice
IČ: 180 51 634 DIČ: CZ18051634
CZECH REPUBLIC

Abstrakt

Tato bakalářská práce popisuje průběh odborné praxe u firmy TRANSEXPRESS Intl. spol. s r.o. Čtenář se dozví základní informace o softwaru Zoneminder, video formátu MJPEG, JavaScriptovém API ResizeObserver a dalších použitých technologiích. V práci je také popsána implementace výkonnější náhrady skriptu zmaudit.pl, což je modul systému Zoneminder, který kontroluje integritu kamerových záznamů. Dále se v práci nachází popis vývoje webového rozhraní umožňujícího sledování obrazu kamer a zaznamenaných událostí systému Zoneminder.

Klíčová slova: Bakalářská práce, odborná praxe, Zoneminder, IP kamery, C#, Mono, ResizeObserver

Abstract

This bachelor thesis describes professional practice at the company TRANSEXPRESS Intl. spol. s r.o. Reader will get basic information about Zoneminder software, MJPEG video format, JavaScript ResizeObserver API and other used technologies. The thesis also describes the implementation of a faster replacement for zmaudit.pl script, which is a Zoneminder system module that checks the integrity of the camera records. The thesis also describes the development of a web interface for viewing camera stream and recorded events of the Zoneminder system.

Key Words: Bachelor's thesis, professional practice, Zoneminder, IP cameras, C#, Mono, ResizeObserver

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
SQL	–	Structured Query Language
CCTV	–	Closed-circuit television
GPL	–	GNU General Public License
CLI	–	Common Language Infrastructure
XML	–	Extensible Markup Language
CSS	–	Cascading Style Sheets
JSON	–	JavaScript Object Notation
AJAX	–	Asynchronous JavaScript and XML
WPF	–	Windows Presentation Foundation
WCF	–	Windows Communication Foundation
DOM	–	Document Object Model
CSV	–	Comma-separated values

Seznam obrázků

1 Úvod

Obsahem této bakalářské práce je zpráva o absolvování individuální odborné praxe kterou jsem vykonal u firmy TRANSEXPRESS Intl. spol. s r.o. U této firmy jsem působil jako brigádník už v době volby témat bakalářských prací a tudíž mě zaujala možnost vykonat bakalářskou práci formou odborné praxe.

V první části této práce se budu zabývat projektem auditu databáze kamerového systému Zoneminder (open-source projekt, jenž umožňuje monitorování, analýzu a ukládání záznamů z CCTV kamerových systémů), který měl za úkol zlepšit výkonnost vestavěného kontrolního skriptu `zmaudit.pl`.

V druhé části se budu věnovat vývoji webové aplikace, jenž by umožňovala sledování obrazu kamer, které jsou na systém Zoneminder připojeny a také sledování uložených záznamů z těchto kamer.

Řešení prvního projektu mi zabralo zhruba 32 pracovních dní, u druhého projektu to bylo okolo 24 dnů. Ostatní práce mi zabrala zhruba 3 dny. Podstatnou část doby řešení mi u obou hlavních projektů zabralo zkoumání chování a vlastností softwaru Zoneminder a to především kvůli nedostatku programátorské dokumentace, která popisuje jen malou část softwaru.

2 O společnosti

Společnost TRANSEXPRESS Intl. spol. s r.o. [1] byla založena v roce 1991. Sídlo má v Ostravě–Heřmanicích a dále má pobočky v Karviné, na Letišti Václava Havla v Praze a na letišti v Mošnově. Firma se zabývá vnitrostátním i mezinárodním zasilatelstvím pomocí silniční nákladní dopravy, letecké, námořní a železniční dopravy. Dále poskytuje služby autoservisu, čerpací stanice, skladování zboží a parkování.

Ve společnosti jsem pracoval na pozici Programátor. Hlavní náplní mojí práce byl vývoj aplikací v jazyce C#, PHP a JavaScript . IT oddělení ve firmě se v současné době skládá z dvou osob. Vyvíjí aplikace pro potřeby firmy, a to jak vnitrofiremní aplikace, tak systémy poskytující služby firemním zákazníkům. Dále spravuje firemní servery a stanice zaměstnanců.

3 Použité technologie a nástroje

Během praxe jsem se setkal s velkým množství softwarových technologií a nástrojů. Některé pro mně byly zcela neznámé, u jiných jsem musel své znalosti prohloubit.

3.1 Zoneminder

Zoneminder [2] je komplexní kamerový bezpečnostní software, který umožňuje zachytávat, analyzovat, ukládat a přehrávat záznamy z bezpečnostních kamer. Je kompletně open–source, zdarma a je vydán pod licencí GPL. Je k dispozici na většinu linuxových systémů, Windows podporovány nejsou.

Podporuje mnoho typů kamer, např. klasické CCTV analogové kamery, IP kamery nebo USB kamery, i kodeků jako H.264, MJPEG a další. [3] Nabízí také API, pomocí kterého je možno plnohodnotně přidávat, nastavovat a odebírat kamery, vyhledávat a mazat záznamy a podobně. [4]

Software také umožňuje u některých typů kamer jejich vzdálené ovládání, přibližování, zaostřování, atd. [5]

Standardně se při všech operacích s daty (přijmutí snímku z kamery, analýza, uložení na disk a vysílání proudu obrazových dat) používá formát JPEG (respektive MJPEG). Tento formát má sice větší paměťovou náročnost (ve srovnání s ostatními formáty používanými pro přenos videa, například H.264), zabírá více místa v úložišti a více zatěžuje síť, ale ušetří se tím výkon procesoru, který by musel vynaložit na kódování do jiného formátu.

3.1.1 Slovník

Některé názvy používáné v Zoneminderu nejsou na první pohled srozumitelné:

- Monitor – Kamera připojená k Zoneminderu.
- Event – Událost, spuštěna po analýze obrazu, obvykle uložení záznamu po rozpoznání změny oproti předchozím snímkům.
- Filter – Pomocí filtru lze nejenom vybírat události, ale i spouštět nad těmito událostmi různé operace, jako je jejich smazání, archivace nebo spuštění nejrůznějších skriptů. Filtry lze také spouštět automaticky na pozadí.
- Zone – Monitory lze seskupovat do oblastí, u kterých lze např. nastavit, že při zachycení pohybu na jednom monitoru začnou nahrávat všechny monitory.

3.1.2 Uložení kamerových záznamů

Data jsou v Zoneminderu uložena na dvou místech, a to v databázi (podporována je MySQL) a v adresářové struktuře na disku. V databázi jsou mimo jiné tyto tabulky:

- Config – Obsahuje základní nastavení Zoneminderu.
- Events – Tabulka obsahující události a informace k nim.
- Filters – Seznam filtrů a jejich vlastností.
- Frames – Obsahuje záznamy o jednotlivých obrázcích s cizím klíčem ke konkrétní události.
- Monitors – Seznam kamer a jejich vlastností.
- Users – Obsahuje seznam uživatelů.

Samotná zaznamenaná kamerová data jsou uložena ve formě JPEG obrázků, uložených v této adresářové struktuře:

`/.../zoneminder/events/{číslo kamery}/` ⋯
⋯ `{rok}/{měsíc}/{den}/{hodina}/{minuta}/{sekunda}/`

Kde `rok`, `měsíc`, `hodina` a `sekunda` se vztahují k začátku události. Složka `{sekunda}/` slouží vždy pro jednu konkrétní událost, kromě samotných JPEG obrázků obsahuje i prázdný soubor pojmenovaný `.{id události}`, id se vztahuje k záznamu v databázi. Kromě těchto složek a souborů se v této adresářové struktuře vyskytují také symbolické odkazy. Ve složce `/.../zoneminder/events/` jsou to odkazy se jménem `nazev_kamery`, které odkazují na adresář s příslušným číslem kamery ve stejné složce. Dále jsou v `/.../zoneminder/events/{číslo kamery}/{rok}/{měsíc}/{den}/` odkazy pojmenované `.{id události}` odkazující na adresář události.

3.1.3 Skript zmaudit.pl

Skript zmaudit.pl [6] je napsaný v jazyku Perl, sloužící k ověření konzistence dat uložených v databázi a v adresářové struktuře v systému souborů. Maže osamocené události, tj. ty které nejsou uloženy na obou místech, např. událost zaznamenaná v databázi, která není uložena v souborovém systému, nebo opačně. Tohoto se využívá například tehdy, když je v nastavení nastavena volba ZM_OPT_FAST_DELETE, která způsobí, že události jsou při mazání Zoneminderem smazány jen z databáze. Skript může běžet v interaktivním i dávkovém režimu. Lze nastavit nejmenší staří události, jenž má skript kontrolovat (volba ZM_AUDIT_MIN_AGE).

3.1.4 Získání aktuálního obrazu kamery

Získat aktuální MJPEG proud Zoneminder monitoru se dá například pomocí jednoduchého http dotazu

`http://{adresa Zoneminder serveru}/cgi-bin` ⋯
⋯ `/nph-zms?mode=jpeg&monitor={id monitoru}&scale={zvětšení}`

Seznam některých parametrů dotazu [7, 8]:

- mode – Nastaví formát dat, jenž se vrátí jako odpověď.
  - single – Jeden JPEG obrázek.
  - raw – Surová obrazová data – kopie dat z paměti. Mime typ image/x-rgb. [10]
  - jpeg – MJPEG stream.
  - zip – Zkomprimovaný formát raw.[1]
- monitor – Vybere monitor (kameru), ze které má získat obraz.
- scale – Nastaví velikost vráceného obrazu. Zadává se v procentech, hodnota `100` znamená originální velikost.
- maxfps – Nastavuje maximální počet snímků za sekundu, jenž má vrácený dotaz

Samotný MJPEG proud se v prohlížeči dá zobrazit pomocí `<img>` tagu, kdy se jako atribut `src` použije HTTP dotaz na MJPEG proud.

3.1.5 Získání záznamu kamery

Zoneminder server poskytuje, pro získání záznamu, podobné rozhraní pro přístup jako při sledování aktuálního obrazu z kamer. Například:

`http://{adresa serveru}/cgi-bin/nph-zms?source=event&event={id události}`

nebo

`http://{adresa serveru}/cgi-bin/nph-zms` ⋯
⋯ `?source=event&monitor={id monitoru}&time={Unix time}`

V dotazu je možné použít stejné parametry jako u dotazu pro aktuální proud z monitorů (viz Sekce 3.1.4). Dále je možné použít tyto parametry [7–9]:

- source – Je–li v dotazu předán parametr source s hodnotu `event`, server vrátí událost místo aktuálního obrazu.
- event – Id události.
- frame – Nastaví obrázek, od kterého se má začít vysílat. Validní hodnota je kladné nenulové celé číslo.
- replay – Ovlivňuje pokračování videa po skončení zvolené události
  - all – Přehraje další událost zvoleného monitoru. Pokud je mezi nimi časová prodleva, vloží mezi ně čekací smyčku (odpočítávání sekund zbývajících do další události)
  - gapless – Přehraje další událost bez čekací smyčky.

[1] Pomocí funkce `compress2` z knihovny zlib

  - single – Po skončení aktuální události zastaví přehrávání.

- rate – Nastavuje rychlost běhu záznamu. Validní hodnota je desetinné číslo, 1 značí standartní rychlost.

- time – Slouží pro vyhledání události podle času. Zadává se ve formátu Unix time.

Konkrétní událost se dá vybrat dvěma způsoby. První možnost je pomocí parametru `event`, jehož hodnota je id události v databázi. Druhá možnost je vybrat kameru pomocí parametru `monitor` a použít parametr `time`, server vrátí první událost příslušného monitoru, jejíž čas ukončení je větší než hodnota parametru `time`.

3.1.6 Ovládání proudu videa

Při sledování zaznamenaných událostí je možné tento MJPEG proud ovládat (pozastavovat, spouštět, zrychlit/zpomalit, přibližovat/oddalovat,...). Klient pošle HTTP POST (viz Výpis 1) nebo GET dotaz obsahující příkaz k příslušné akci. Server poté upraví proud a pošle klientovi odpověď ve formátu JSON (viz Výpis 2).

connkey – náhodně generovaný klíč určující konkrétní MJPEG proud

```
function pause( key )
{
   var httpPost = new XMLHttpRequest();
   httpPost.open("POST","http://server/zm/index.php",true);
   httpPost.setRequestHeader(
           "Content-type",
           "application/x-www-form-urlencoded");
   httpPost.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      console.log(this.responseText);
    }};
   xhttp.send("view=request&connkey=" + key + "&request=stream&command=1");
}
```

Výpis 1: Příklad POST dotazu – tento dotaz pozastaví proud záznamu

```
{
"result":"Ok",
"status":
   {
      "type":3,
      "event":132456798,
```

```
      "progress":5,
      "rate":1,
      "zoom":"1.0",
      "paused":0
   }
}
```

Výpis 2: JSON odpověď na dotaz položený ve Výpisu 1

Druh příkazu je specifikován v parametru `command`. Jsou možné tyto operace pauza (hodnota parametru `command` je `1`), spustit přehrávání (`2`), stop (`3`), zrychlení (`4`), posun o snímek dopředu (`5`), posun s snímek dozadu (`6`), zpomalení (`7`), přiblížení (`8`), oddálení (`9`), ukončení přenosu (`17`) nebo jen dotaz na aktuální stav (`99`). [11]

3.1.7 Zoneminder REST API

Zoneminder nabízí REST API [4] pomocí něhož se dají zjišťovat seznamy monitorů, událostí a upravovat jejich vlastnosti. Odpovědi ze serveru přicházení ve formátu JSON nebo XML (dle použité koncovky). Toto API v aplikaci používám pro získávání informací o událostech (viz sekce 5.3).

Příklad:
`http://server/zm/api/monitors.json`
– seznam všech monitorů a informace o nich
`http://server/zm/api/events/index/MonitorId:1.json`
– seznam událostí náležících k monitoru 1
`http://server/zm/api/events/index/MonitorId` ⋯
⋯ `:1/StartTime >=:2018-01-01 10:12:13/EndTime <=:2018-01-02 11:12:23.json`
– seznam událostí náležících k monitoru 1 omezený příslušným časovým oknem.

3.2 ResizeObserver

ResizeObserver je JavaScriptové API, jenž umožnuje pomocí událostí[2] (viz příklad Výpis 3) reagovat na změny velikostí prvnků. Nevýhoda tohoto API je v jeho (ne)podpoře prohlížeči, v době psaní této bakalářské práce je podporován jen v prohlížečích Chrome (desktop i Android verze), Opera a Vivaldi. Vzhledem k tomu, že aplikace je vyvíjena přednostně pro Chrome, tak to není podstatné omezení.

Toto API má poměrně malou náročnost na systémové prostředky. Je voláno (respektive definovaná funkce je volána) přímo prohlížečem při změně velikosti sledovaného prvku. Vzhledem k tomu, že prohlížeč musí sledovat prvky i bez aplikace tohoto API (kvůli aplikování CSS

[2] Ve smyslu „eventů“ v programovacích jazyce, zde v JavaScriptu.

pravidel, zvýrazňování,...), volání funkce pro nastavení správného poměru podstatně neovlivní množství spotřebovaných systémových prostředků. [12]

```
funtion OnLoad()
{
   const resizeObserver = new ResizeObserver(entries =>
    {
        for(let entry of entries)
        {
            var height = entry.contentRect.height;
            entry.target.style.margin = (height * 1.5)+"px";
        }
    });
    var elements = document.querySelectorAll(".someClass");
    elements.forEach(function (element)
      {
         resizeObserver.observe(element);
      });
}
```

Výpis 3: Příklad použití ResizeObserver API

3.3 MJPEG

Motion JPEG (zkráceně M-JPEG nebo MJPEG) [13], je video formát u něhož je každý snímek tvořen JPEG obrázkem. Jeho nevýhodou je poměrně velký datový tok (v porovnání s ostatními formáty používanými na přenos videa) a taky to, že není standardizovaný. Jelikož je to ale jenom proud za sebou jdoucích JPEG snímků (které jsou standardizovány), tak nebývá problém s nekompatibilitou jednotlivých implementací.

Je podporován nástroji balíku FFmpeg (kódování, dekódování) a prohlížeči Chrome, Opera, Firefox a Edge. Existuje také Java applet Cambozola [14], jenž umožňuje sledovat MJPEG proud v prohlížečích, které standardně nemají jeho podporu. Je často využíván (IP) kamerami.

3.4 Mono

Mono je open–source projekt implementující interpret CLI bytekódu, překladač jazyka C# a většinu knihoven .NET Frameworku.

V době psaní této bakalářské práce podporuje C# verze 7.0, je kompatibilní s .NET Frameworkem 4.7 (kromě WPF a části WCF) a podporuje hostování ASP.NET. [15] Mono je k dispozici na velké množství platforem, například x86, x64, ARM, MIPS nebo PowerPC. Na

linuxových systémech je k dispozici v repozitářích většiny hlavních distribucí, popř. jsou k dispozici repositáře od tvůrců, které obsahují nejnovější verze.

3.5 Git

Git je open–source verzovací systém vytvořený Linusem Torvaldsem původně pro správu a spolupráci při vývoji jádra Linuxu. Je to v dnešní době jeden z nejrozšířenějších verzovacích systémů. Lze ho ovládat z příkazové řádky, pomocí programů s grafickým rozhraním, případně pomocí vestavěné komponenty ve Visual Studiu.

3.6 PHP

PHP je objektový skriptovací programovací jazyk, určený převážně pro vývoj webových aplikací. Interprety jazyka jsou k dispozici pro většinu linuxových distribucí nebo Windows systémů. Je součástí (případně lze jeho podporu doinstalovat) většiny webových serverů jako například Apache, Nginx nebo IIS [16]. Je to typově dynamický jazyk, v určitých případech ale lze typ nastavit (například parametry funkce nebo návratová hodnota funkce).

Pro PHP existuje rozšíření Xdebug, jenž umožňuje krokování, umisťování breakpointů, čtení hodnot proměnných a další.

3.7 PHP Tools for Visual Studio

PHP Tools for Visual Studio je komerční plugin pro Microsoft Visual Studio vyvinutý firmou Devsense. Přidává podporu vývoje v PHP, obsahuje integrovaný PHP server. Podporuje napojení na vzdálený webový server a Xdebug.

3.8 Webserver Apache

Apache je open–source webový server v současnosti dostupný na linuxových distribucích a Windows. Podporuje mnoho různých jazyků, pro které může zpracovávat dotazy, lze do něj doinstalovat módy (rozšíření) například pro PHP, ASP.NET a další.

3.9 WMware ESXi

WMware ESXi je operační systém a hypervizor. Je to proprietární software vyvinutý firmou WMware. Je určený pro platformu x86 (respektive x64). Dokáže hostovat velké množství virtuálních strojů. Pokud je použit nadřazený vCenter Server, dokáže být propojený s ostatníma ESXi serverama a podporovat funkce jako centrální správa všech strojů, High Availability (téměř nepřetržitá dostupnost virtuálního stroje i při poruše hypervizoru).

Mezi jeho nevýhody patří špatná podpora některých hardwarových zařízení, hlavně řadičů RAID polí a síťových karet.

3.10 AJAX

AJAX [17] je popis způsobu, jakým může webová aplikace asynchronně komunikovat pomocí HTTP dotazů a JavaScriptu. Aplikace může například získat nová data i bez opakovaného kompletního načtení stránky (viz příklad Výpis 4).

```
function fce( )
{
   var request = new XMLHttpRequest();
   request.open("GET","http://server/somePath/",true);
   request.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("something").innerText=this.responseText;
    }};
   request.send();
}
```

Výpis 4: Příklad použití AJAX

3.11 CSS

Cascading Style Sheets (CSS), neboli kaskádové styly, slouží k upravě vzhledu prezentace vytvořené pomocí značkovacího jazyka, obvykle HTML. Současná verze CSS3 umí při správném použití pružně reagovat na různé velikosti zobrazovacích zařízení.

4 Audit databáze Zoneminderu

Cílem tohoto projektu bylo zkrátit dlouhou dobu běhu skriptu `zmaudit.pl` (viz sekce 3.1.3) softwaru Zoneminder. Hlavní úkol této komponenty je kontrola konzistence dat v databázi a adresářové struktuře. Vzhledem k velkému počtu malých souborů, jenž musí skript projít, jde o časově velice náročnou operaci. Při množství kamerových záznamů uložených na firemních serverech (5 až 10 TB na server) by doba běhu tohoto skriptu byla stovky hodin. Bylo mi zadáno zkrátit dobu běhu tohoto skriptu na přijatelnou hodnotu.

Vzhledem k velkému množství souborů, které musí program projít (viz Sekce 3.1.2) byl problém s výkonností hlavně u přístupu k disku, procesor ani RAM paměť úzkým hrdlem nebyly. Proto jsem se při implementaci zaměřil hlavně na omezení přístupu na disk i za cenu zvýšení procesorové a paměťové náročnosti.

4.1 Použité technologie

Kvůli svým nulovým zkušenostem s jazykem Perl jsem se rozhodl tento skript neupravovat, ale napsat od znova program, který by ho nahradil.

Vzhledem k tomu, že rozdíl rychlostí běhu programů a skriptů napsaných pro tento problém v různých jazycích (majících potřebné vlastnosti, například C++, Bash, PHP), by byl kvůli nenáročnosti na CPU zanedbatelný, zvolil jsem především kvůli rychlosti vývoje .NET Framework a jazyk C#, jehož assembly bude na linuxovém serveru spouštěná pomocí platformy Mono.

K vývoji jsem použil vývojové prostředí Microsoft Visual Studio 2015. Při optimalizaci výkonu jsem používal profiler, který je součástí platformy Mono.

4.2 Testovací server

Pro potřeby testování jsem si vytvořil vývojovou verzi serveru se softwarem Zoneminder. Místo duplikace některého z již existujících serverů jsem se rozhodl vytvořit tento vývojový server nainstalováním čistého operačního systému, softwaru Zoneminder, MySQL databáze (se kterou Zoneminder pracuje), platformy Mono a další potřebných programů.

Tento postup jsem zvolil převážně proto, že jsem se Zonemindrem neměl předchozí zkušenosti a chtěl jsem se zorientovat v jeho funkcích a komponentách.

Server jsem nainstaloval jako virtuální stroj. Jako host mi posloužil virtualizační server s WMware ESXi hypervizorem. Operačním systémem pro Zoneminder server jsem zvolil Ubuntu Server 16.04 LTS.

4.3 Testování rychlosti souborových systémů

Jako jeden ze způsobů jak zrychlit přístup k disku, jsem se rozhodl vybrat správný systém souborů pro oddíl s kamerovými záznamy. Napsal jsem si testovací program, jenž měl za úkol navštívit všechny složky událostí a měřit čas, který k tomu potřeboval.

4.3.1 Testované souborové systémy

Testoval jsem tyto souborové systémy:

- ext4
- ReiserFS
- Btrfs
- Btrfs - zapnuty volby *nodatasum* a *nodatacow* [18]
- Btrfs - zapnuta volba *compress=lzo* [18]

Volba *nodatacow* vypne vlastnost copy-on-write pro nově vytvořené soubory, *nodatasum* vypne kontrolní součet pro nově vytvořené soubory. Tyto dvě volby jsou na sobě závislé, vypnutí jedné vypne i druhou. Volba *compress=lzo* zapne komprimaci nově vytvořených souborů pomocí algoritmu LZO (Lempel–Ziv–Oberhumer).

4.3.2 Výsledky

První testy jsem provedl bez zátěže, tj. Zoneminder byl vypnutý a na serveru běžely jenom standardní systémové procesy (viz Obrázek 1).

Další testy jsem provedl se zátěží, tj. Zoneminder byl zapnutý a server byl zatížen jako při normálním běhu. S tímto nastavením jsem netestoval ReiserFS vzhledem k jeho velice špatným výsledkům v předchozím testu (viz Obrázek 2).

V testu nejlépe dopadl souborový systém ext4, bude tudíž i nadále používán pro oddíly s adresářovou strukturou programu Zoneminder.

4.4 Implementace náhrady skriptu zmaudit.pl

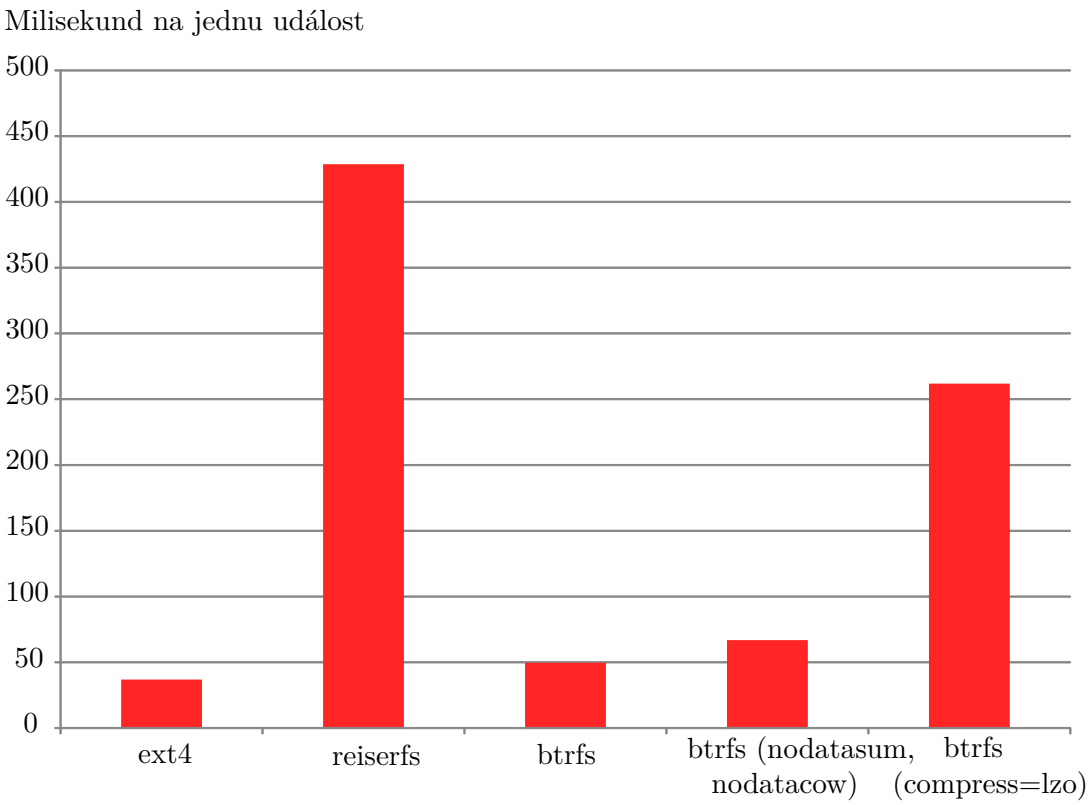
Při implementaci samotného programu (náhrada skriptu `zmaudit.pl`, viz 3.1.3) jsem použil velkou část z testovacího programu (viz sekce 4.3), jehož třídy jsem psal univerzálněji s tím, že je poté znovu použiji.

Program je napsán jako konzolová aplikace, nemá tudíž grafické uživatelské rozhraní. Lze jej spouštět uživatelsky nebo pomocí skriptů, popř. naplánovaných úloh.

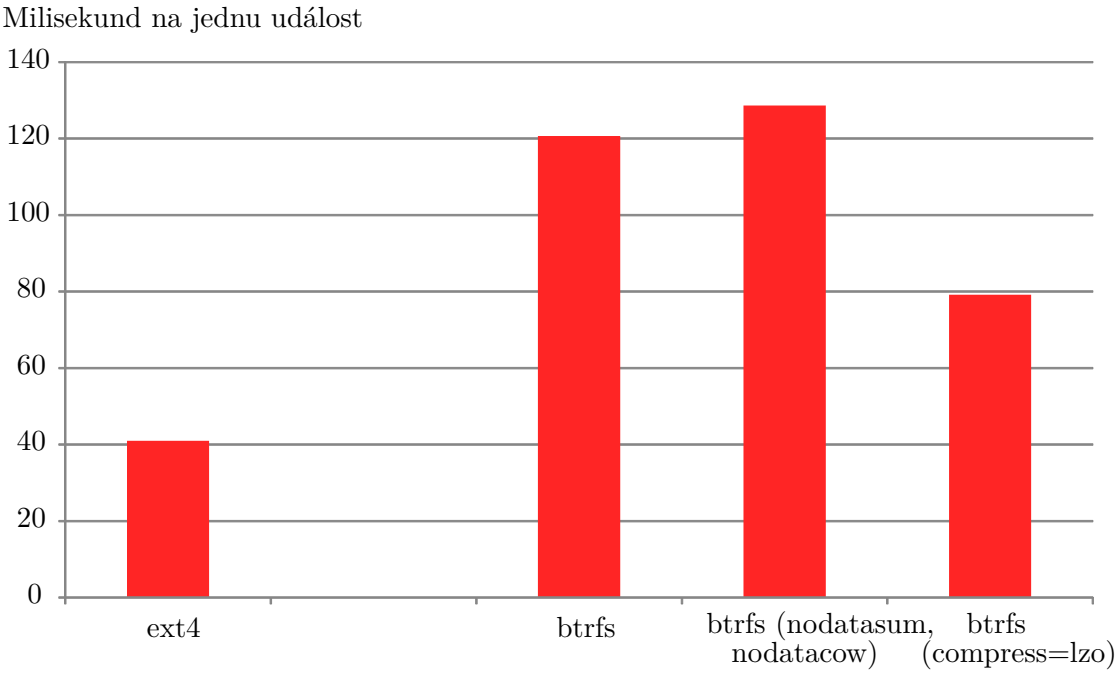
Lze jej nastavit pomocí parametrů programu anebo konfiguračního souboru. Zadané parametry mají přednost před konfiguračním souborem. Pokud není konkrétní parametr zadán a nenachází se ani v konfiguračním souboru, je použita výchozí hodnota zadaná přímo v programu. Konfigurační soubor je představován XML souborem `App.config` a je čten pomocí třídy ConfigurationManager[3].

Program je psán s ohledem na to, že nejvíce času stráví čekáním na data z diskového pole. Při jeho běhu jsou spuštěna tři vlákna, kdy jedno má na starosti procházení dat v souborovém

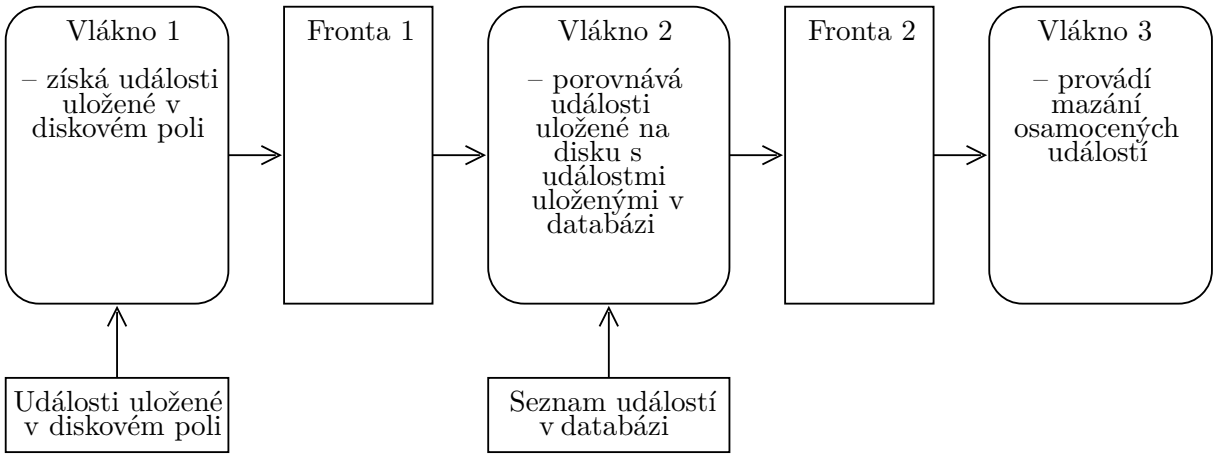
[3] System.Configuration.ConfigurationManager



Obrázek 1: Graf výsledků testování rychlosti souborových systémů – testování bez zatížení diskového pole



Obrázek 2: Graf výsledků testování rychlosti souborových systémů – testování se zatíženým diskovým polem



Obrázek 3: Rozdělení práce programu na tři vlákna

systému a uložení nalezených událostí do fronty[4], odkud si je bere druhé vlákno a porovnává je se záznamem v databázi. Pokud se událost nenachází na obou místech (na disku i v databázi), tak ho označí ke smazání (buďto z databáze nebo z disku) a vloží do další fronty. Z této poslední fronty si ho vezme poslední vlákno, které s ním provede požadovanou operaci (smazání z disku nebo z databáze). Toto vlákno může, podle nastavení, skutečně vykonat tuto operaci nebo ji jenom fingovat (nicnedělám nebo vypsáním informací do log souboru). Rozdělení na vlákna je znázorněno na Obrázku 3.

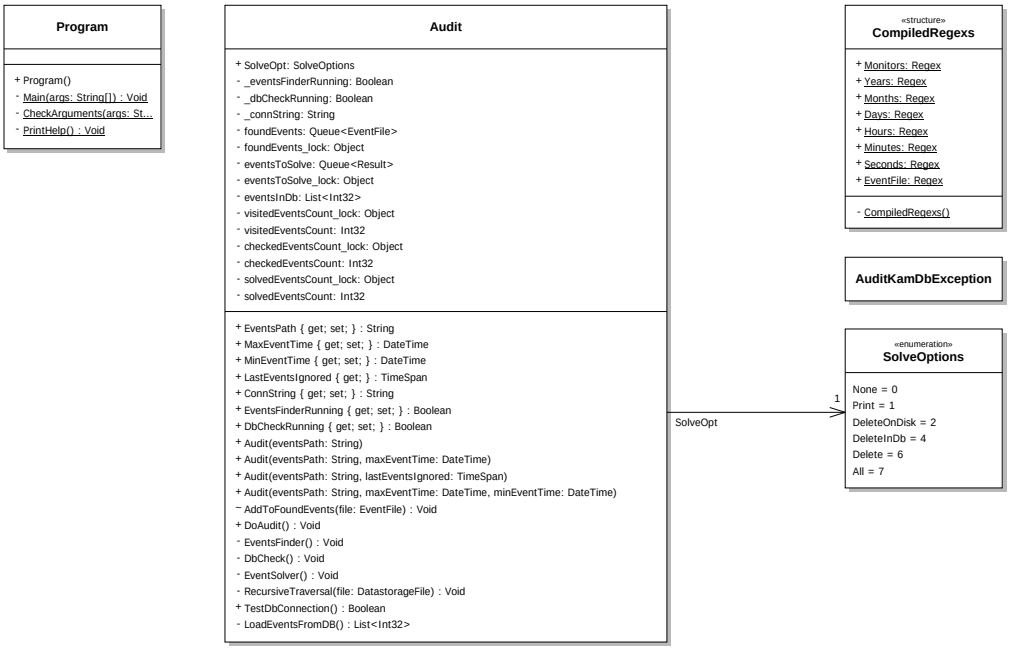
Získání seznamu událostí z databáze je prováděno hromadně při startu druhého vlákna. Není tedy pokládán dotaz na databázi pokaždé při kontrole nalezené události, ale je jen prohledána datová struktura v paměti, což velice zrychluje běh programu.

Toto řešení a také to, že mezi nalezením události a jejím zpracováním (porovnáním) je přestávka (způsobená čekáním událostí ve frontě) má tu nevýhodu, že seznam událostí může být v době porovnávání již neaktuální. Vzhledem k tomu, že při procházení adresářové struktury na disku se ignorují události (resp. celé podstromy s událostmi) mladší než určený časový údaj[5], tak jediný chybný stav, který může nastat je, když událost, která měla být smazána, už smazána je. Její mazání tedy program neprovede a pokračuje dál.

Program se skládá z několika tříd. Třída `Program`(viz obrázek 4) obsahuje kromě vstupního bodu aplikace (metoda Main(...)) i metody sloužící ke čtení vstupních parametrů a konfiguračního souboru a vypsání případné nápovědy. Po zpracování konfigurace třída `Program` vytvoří objekt třídy `Audit`, nastaví mu příslušné hodnoty dle konfigurace a spustí samotný audit.

Třída `Audit`(viz obrázek 4) obsahuje hlavní funkčnost programu, vytváří vlákna a obsahuje fronty čekajících událostí. V každém ze tří vláken je spuštěna privátní metoda třídy `Audit`, která provádí úkol daného vlákna, stará se o ukládání nebo načítání událostí z front a podobně. Jsou

[4]Použita standartní .NET fronta System.Collections.Generic.Queue<T>
[5]Standartně doba startu programu mínus 30 minut – lze nastavit



Obrázek 4: Třídní diagram programu – první část

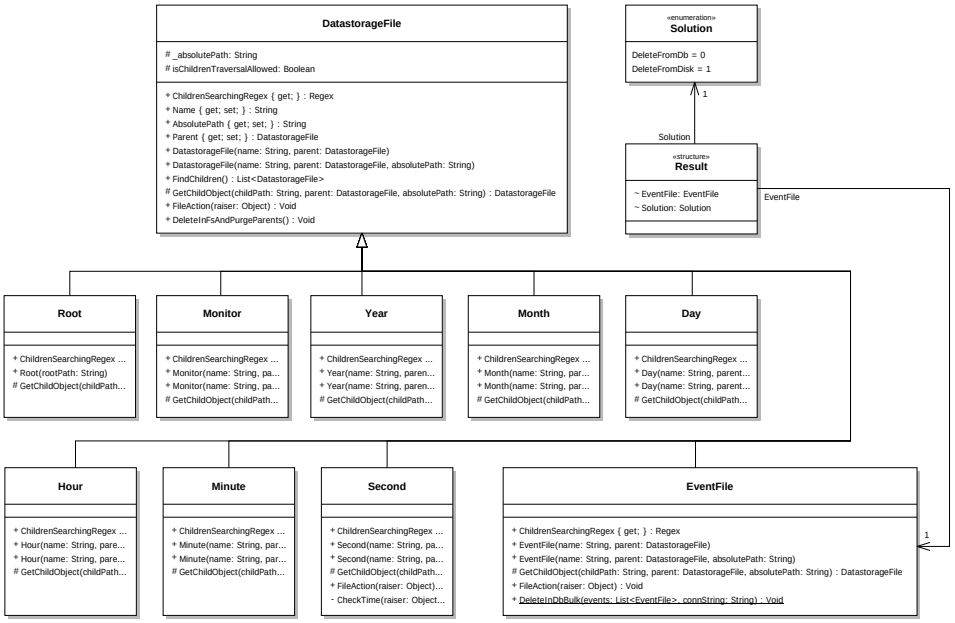
to metody `EventsFinder()` pro první vlákno, `DbCheck()` pro druhé vlákno a `EventSolver()` pro třetí vlákno.

Prohledávání souborového systému v prvním vlákně se provádí jednoduchou rekurzivní metodou (`RecursiveTraversal(DatastorageFile file), viz Obrázek 4`), která navštíví daný soubor[6] reprezentovaný objektem třídy DatastorageFile(ve skutečnosti je to ale jeden z potomků této třídy, viz Obrázek 5) a zavolá znovu tuto rekurzivní funkci na příslušné podsložky. Při „navštívení“ souboru[6] se zavolá virtuální metoda `FileAction()` navštíveného objektu `DatastorageFile`. Tato metoda ve třídě `DatastorageFile` neobsahuje žádnou akci, ale někteří jeho potomci tuto metodu překrývají. V tom případě je pak zavolána tato překrývající metoda, která vykoná akci náležející k příslušnému objektu.

Například u třídy `Second` (viz Obrázek 5) je z názvů nadřazených složek (každý potomek třídy `DatastorageFile` si uchovává referenci na objekt, který představuje jeho nadřazenou složku) sestaven čas vytvoření události (viz názvy složek, Sekce 3.1.2) a podle tohoto údaje a nastaveného minimálního a maximálního času kontrolovaných událostí jsou prohledávány jenom vyhovující podsložky. U třídy `EventFile` (viz Obrázek 5) přetížená metoda `FileAction()` volá metody související s navštívením souboru události (viz Sekce 3.1.2), například přidání nalezené události do druhé fronty.

Po tom co rekurzivní metoda „navštíví“ daný soubor[6], tj. zavolá metodu`FileAction()` objektu reprezentujícího daný soubor[6], zavolá metodu `FindChildren()` daného objektu. Pokud

[6]Souborem je v této kapitole myšlen jak soubor tak i složka. V adresářové struktuře, ve které jsou uloženy události, jsou kromě posledního souboru všechno složky (viz Sekce 3.1.2).



Obrázek 5: Třídní diagram programu – druhá část

objekt reprezentuje složku (vlastnost `isChildrenTraversalAllowed` je nastavena na `true`), tak tato metoda vrátí seznam souborů[6] nacházející se v dané složce. Tento seznam obsahuje objekty třídy `DatastorageFile` (respektive některého z jeho potomků).

Potomci (viz Obrázek 5) třídy `DatastorageFile` odpovídají adresářové struktuře, ve které jsou uloženy události v souborovém systému (viz sekce 3.1.2). Tyto třídy si udržují referenci na objekt reprezentující nadřazenou složku (kromě objektu třídy `Root`, jenž reprezentuje hlavní složku úložiště událostí). Tyto třídy také překrývají metodu `GetChildObject(...)`, která vrací objekt jiného potomka třídy `DatastorageFile` tak, jak to odpovídá adresářové struktuře (například třída `Month` vrací objekt třídy `Day`).

Soubory[6], vrácené metodou `FindChildren()` jsou filtrovány pomocí regulárních výrazů tak, aby byly vráceny jenom vhodné soubory[6]. Ve složkách se totiž kromě těchto souborů mohou vyskytovat například symbolické odkazy, nebo i soubory[6] nijak nesouvisející s událostmi Zoneminderu.

Tyto regulární výrazy jsou všechny umístěny ve struktuře `CompiledRegexs` (viz Obrázek 4). Původně byly umístěny v jednotlivých třídách, ve kterých se používají, ale při optimalizaci programu jsem za pomoci profileru zjistil, že toto filtrování pomocí regulárních výrazů zabírá poměrně velký výpočetní čas a rozhodl jsem se ho zrychlit. Toho jsem docílil několika způsoby. Jednak byla při vytváření objektů třídy Regex[7] použita volba `RegexOptions.Compiled`, která způsobí, že tyto objekty jsou zakompilovány do assembly, což přinese podstatné zrychlení při jejich mnohonásobném použití. Dále jsem je všechny umístil do statické struktury `CompiledRegexs`,

[7]System.Text.RegularExpressions.Regex

což má v tomto případě za následek, vzhledem k chování kompilátoru Mono Runtime [19], zrychlení přístupu do paměti k těmto objektům.

4.5 Výsledek

Běh programu, tj. kontrola všech dat (událostí), zabrala mezi 7 až 9 hodinami, což je oproti předchozím stovkám hodin podstatné zlepšení. Tato doba je již dostatečná pro použití v ostrém nasazení, proto byl tímto programem na firemních serverech nahrazen standardní skript *zmaudit.pl*.

5 Webová aplikace pro sledování obrazu a záznamu kamer systému Zoneminder

Druhým úkolem, který mi byl zadán, byl vývoj webové aplikace, ve které bude možno sledovat v reálném čase obraz z IP kamer („monitorů“) a také sledovat záznamy z těchto kamer.

Tato aplikace má za úkol nahradit stávající firemní aplikaci, která umožňuje sledování obrazu, ale ne záznamu a nemá přizpůsobení pro displeje s malým nebo velkým rozlišením a má přílišnou náročnost na systémové prostředky (hlavně na procesor a síť).

Aplikace je určena na použití na vnitřní firemní síti. Přístup k ní mají jen určení zaměstnanci a zároveň každý z těchto zaměstnanců nemusí mít přístup ke všem kamerám. Jen část těchto zaměstnanců má možnost kromě aktuálního obrazu sledovat také zaznamenané události.

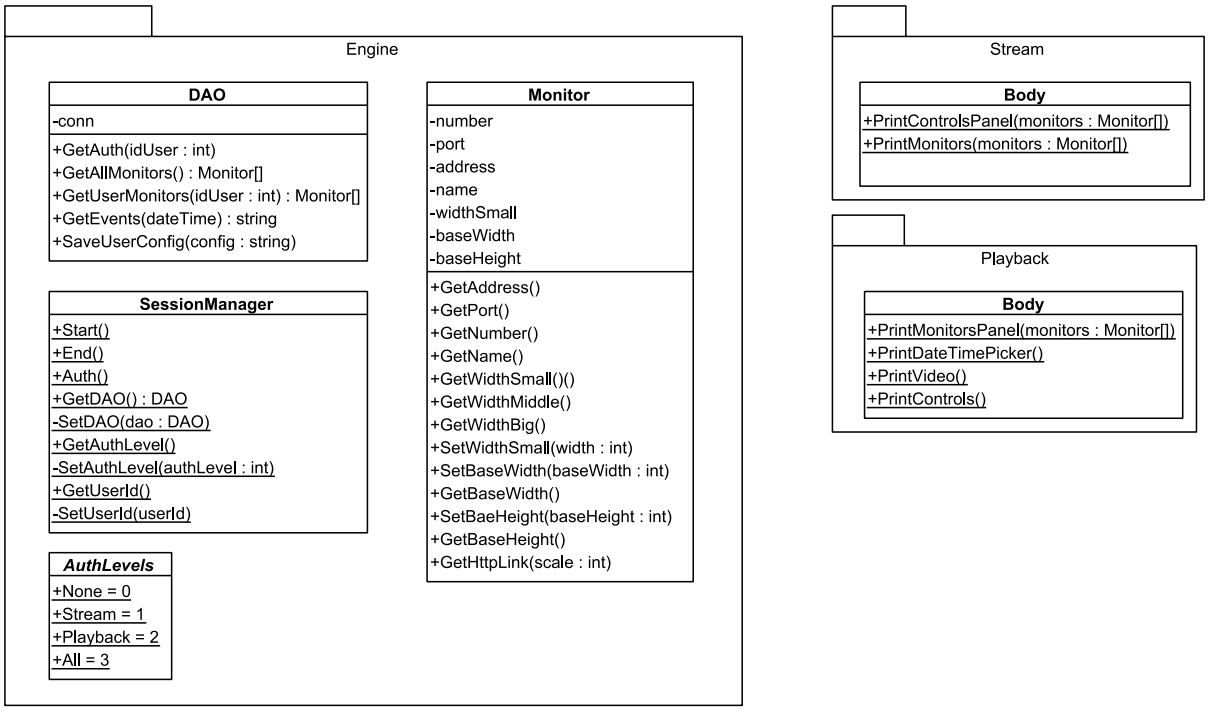
Požadavek na kompatibilitu je omezen na prohlížeč Google Chrome, který se ve firmě standartě používá spolu s Internet Explorerem. Kompatibilita s ním ale není vyžadována, jelikož nativně nepodporuje MJPEG formát (viz Sekce 3.3) a tudíž by bylo potřeba použít JAVA applet nebo použít experimentální verzi Zoneminderu, která podporuje proud videa s kodekem H.264.

Aplikace je rozdělena na dvě části – serverovou a klientskou část. Serverová část je napsaná v jazyce PHP (verze 7.1). Klientská část je naprogramována v jazyce JavaScript, vzhled je řešen pomocí HTML a CSS. Při vývoji JavaScriptových a PHP částí aplikací nebyly použity nadstavby nebo frameworky (jako Angular.js, React.js, Nette,...). Jako webový server je použit Apache. Data jsou uložena v několika MySQL databázích.

5.1 Popis serverové části aplikace

Serverová část aplikace se skládá z třídy `SessionManager` (viz Obrázek 6), jenž obsahuje statické metody komunikující se superglobální proměnou (přesněji polem) `$_SESSION`. Kromě toho také zajišťuje autentizaci uživatele funkcí `Auth()`, která si ověří identitu uživatele (pomocí předpřipravené firemní knihovny) a vytvoří objekt třídy `DAO` (viz Obrázek 6) a uloží ho do proměnné `$_SESSION`. Třída `DAO` se stará o práci s daty, a to jak o přímou práci s databází, tak i komunikaci s REST API Zoneminderu (viz Sekce 3.1.7). Objekty třídy `Monitor` (viz Obrázek 6) slouží jako nosiče dat a také obsahují funkci ke zformátování výstupního dotazu pro MJPEG video proud (viz Sekce 3.1.4). Tyto objekty vytváří objekt třídy `DAO`, jenž údaje k nim získá z databáze. Abstraktní třída `AuthLevels` (viz Obrázek 6) obsahuje pouze konstanty určující možnosti sledování aktuálního obrazu anebo záznamu. Takto použitá třída nahrazuje typ `enum` známý z různých jiných programovacích jazyků (například C#, C++, Java), jelikož v jazyce PHP typ `enum` ani nic co by ho zastupovalo neexistuje.

Tyto výše zmíněné třídy se nacházejí ve jmenném prostoru („namespace“) s názvem `Engine`, jenž se používá pro obě části aplikace (sledování aktuálního obrazu i záznamu). Dále existují



Obrázek 6: Třídní diagram serverové části webové aplikace pro sledování kamer Zoneminderu

jmenné prostory `Stream`, používající se pro část aplikace pro aktuální obrazové proudy z kamer, a `Playback`, jenž se používá v části pro sledování záznamu.

5.2 Sledování aktuálního obrazu kamer

Webová stránka (viz Obrázek 7 a 8) je rozdělena na pruh s nastavením videa z kamer a na zobrazení samotného obrazu z kamer. U kamer lze nastavit velikost obrazu (po třech krocích), popřípadě jestli se kamera má vůbec zobrazovat a posouvat pořadí kamer. Toto nastavení se ukládá do databáze a při dalším spuštění aplikace je obnoveno napříč jeho zařízeními, výjimkou je zobrazení na mobilních zařízeních s malými displeji, kde jsou ze začátku všechny kamery vypnuty a uživatel si vybere kterou kameru nebo kamery zobrazí.

Správné zarovnání prvků (elementů) s kamerami u prvků se stejně nastaveným krokem velikosti (nejmenší, střední, největší) je provedeno pomocí jejich stejné fyzické velikosti. U prvků, u kterých je nastaven jiný krok velikosti, je tento krok vždy celočíselný násobek o krok menší velikosti.

Prvek s obrazem z kamery (přesněji `<div>`, který obaluje samotný `<img>` prvek) si uchovává poměr stran 16 ku 9. Toho jsem chtěl původně provést čistě pomocí CSS (hlavně z výkonnostních důvodů), nenašel jsem ale možnost jak toho docílit, pokud není předem známý alespoň jeden rozměr MJPEG proudu (respektive prvku `<img>` do kterého je umístěn). Ten znám není, protože se při běhu dynamicky mění v závislosti na parametru `scale` v http dotazu. Z tohoto důvodu jsem využil JavaScriptové API ResizeObserver (viz Sekce 3.2). Za pomoci tohoto API sleduji

změny velikostí těchto prvků a upravuji nejenom jejich poměr stran, ale i hodnotu parametru `scale` v http dotazu. Tuto hodnotu vypočítávám tak, aby výsledný proud dat měl rozlišení vhodné pro umístění do `<img>` prvku, tj. aby rozlišení nebylo příliš velké (zbytečně spotřebovává systémové prostředky, hlavně procesor a síť) nebo příliš malé (špatná kvalita obrazu).

Posunování kamer je prováděno pomocí přemisťování potomků v DOM reprezentaci HTML stránky (viz Výpis 5). Toto řešení, což je v podstatě jenom změna referencí u několika DOM objektů, se vyznačuje větší rychlostí (respektive nároky na procesorový čas) v porovnání s jinými řešeními (jako například změny proměnné `innerHTML` u DOM objektu, což si vyžádá přebudování celého DOM stromu).

```
function PositionInc(elementName)
{
    var element = document.getElementById(elementName);
    if (element === null) return; //pokud prvek neexistuje tak funkce skonci
    var next = element.nextElementSibling;
    if (next === null)
        element.parentElement.insertBefore(element, element.parentElement.
            firstElementChild); //pokud dalsi potomek neexistuje, tak skoc na
            zacatek
    else
    {
        next = next.nextElementSibling;
        element.parentElement.insertBefore(element, next);
    }
}
```

Výpis 5: Kód v JavaScriptu – přehození pozice s dalším DOM prvkem

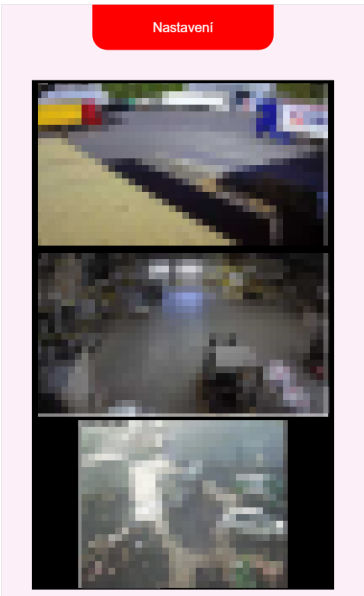
5.3 Sledování záznamu

Druhá část této webové aplikace umožňuje sledování kamerami zaznamenaných událostí. Webová stránka (viz Obrázek 9) je rozdělena na vrchní část, kde jsou umístěny tlačítka na přepínání mezi různými monitory.

V prostřední části se nachází pole pro výběr data a času[8], tlačítko pro vyhledání tohoto výběru a seznam vyhledaných událostí. Tento seznam[9] obsahuje jednadvacet položek, kdy jedenáctá položka je nejbližší s časem větším než čas hledané události. Ostatní položky jsou předcházející nebo následující po této položce, seřazeno dle času od nejmenšího. Při kliknutí na jednu z možností seznamu se začne přehrávat konkrétní událost. Po kliknutí na tlačítko pro

[8] HTML prvek `input` typu `datetime-local`.
[9] HTML prvek `select`

Obrázek 7: Vzhled rozhraní – sledování kamer v reálném čase



Obrázek 8: Vzhled rozhraní na mobilním zařízení – sledování kamer v reálném čase

Obrázek 9: Vzhled rozhraní – sledování záznamu kamer

vyhledání se automaticky spustí ona nejbližší událost (jedenáctá položka). Vpravo od těchto prvků se nachází prvek se samotným videozáznamem události.

Pod tímto prvkem se nachází tlačítka, ovládající průběh tohoto videa a to tlačítka pozastavit, přehrát, předchozí a další událost, zrychlení a zpomalení přehrávání a posun o jeden snímek vpřed a vzad. Tlačítka zrychlení a zpomalení je možné použít jen při spuštěném přehrávání, naopak tlačítka posunu o jeden snímek lze použít jen při pozastaveném přehrávání.

6 Ostatní projekty

V rámci praxe jsem pracoval i na několika menších projektech, které mi nezabraly mnoho času.

6.1 Získání dat z SQL CE databáze a jejich zformátování

V rámci tohoto projektu bylo potřeba získat data z Microsoft SQL CE (SQL Server Compact) databáze.

Tato databáze je používána aplikací Clonet, což je produkt firmy NZ SERVIS, který sleduje několik určených složek a čeká na XML soubory, jenž do nich budou vloženy. Tyto XML soubory poté zpracuje a podle informací v nich uvedených komunikuje, přes servery VAN operátora, s Celní správou ČR. Informace o této komunikaci poté ukládá do výše zmíněné databáze.

Můj program, napsaný v jazyce C#, běží na serveru jako služba. Pomocí třídy FileSystemWatcher[10], sleduje složky, do kterých jsou ukládány ony XML dokumenty. Pokud zaregistruje vložení příslušného souboru, tak po chvíli (100ms) čekání otevře databázi a načte si z ní požadované údaje o přenosu. Tyto údaje posléze naformátuje do do tvaru HTML tabulky a uloží jako soubor do sdíleného úložiště. Tento soubor poté načítá webový server a zobrazí jej jako část webové stránky.

6.2 Čtení tabulek z PDF

V tomto projektu jsem měl za úkol napsat program, který přečte specifické informace z PDF souboru a tyto informace zformátuje do formátu CSV. V těchto souborech, což bylo několik stovek faktur, byly v tabulce uloženy informace o různých produktech, ceny, váhy a další. Vzhledem k formátu PDF souboru, jsou samotné informace uloženy jako řetězec znaků o určitých souřadnicích.

Napsal jsem, v jazyce C# a za pomoci knihovny iTextSharp[11], konzolovou aplikaci, která četla určitým stylem naformátované PDF dokumenty a získávala z nich požadované data. Nad těmito daty posléze provedla požadované operace a uložila je do CSV souborů.

[10] System.IO.FileSystemWatcher

[11] Knihovna pro platformu .NET, umožňující čtení, vytváření a upravování PDF dokumentů.

7 Závěr

Při absolvování odborné praxe jsem využil velké množství znalostí získaných při studiu na vysoké škole, a to především znalosti získané v těchto předmětech:

- Programování I a II
- Algoritmy I a II
- Úvod do databázových systémů, Databázové a informační systémy
- Úvod do softwarového inženýrství
- Programovací jazyky I a II
- Uživatelské rozhraní
- Telekomunikační sítě, Počítačové sítě
- Architektura technologie .NET
- Seminář z programování
- Vývoj informačních systémů

7.1 Znalosti a dovednosti scházející v průběhu odborné praxe

V průběhu odborné praxe mi scházely především specifické znalosti systému Zoneminder. Z obecnějších znalostí jsem se musel doučit především práci s GITem a jazyky JavaScript a PHP.

7.2 Dosažené výsledky v průběhu odborné praxe a celkové hodnocení

V první části praxe jsem implementoval program pro kontrolu úložiště pro systém Zoneminder. Tento program podstatně zkrátil dobu kontroly, čímž naplnil zadání. V druhé části jsem vyvinul webovou aplikaci, která umožňuje přístup ke kamerovým záznamům. Odbornou praxi považuji za dobrou zkušenost, při které jsem se mnohé naučil.

Literatura

[1] TRANSEXPRESS – O nás. TRANSEXPRESS [online]. Ostrava: TRANSEXPRESS Intl. spol. s r.o., 2018 [cit. 2018-03-24]. Dostupné z: http://transexpress.cz/cz/o-nas/

[2] ZoneMinder. [online]: ZoneMinder, 2018 [cit. 2018-04-01]. Dostupné z: https://zoneminder.com

[3] User Guide – Zonemidner Documentation. [online]: Zoneminder, 2014 [cit. 2018-03-25]. Dostupné z: https://zoneminder.readthedocs.io/en/latest/userguide/index.html

[4] API – ZoneMinder documentation. [online]: Zoneminder, 2014 [cit. 2018-03-25]. Dostupné z: http://zoneminder.readthedocs.io/en/stable/api.html

[5] Camera Control – ZoneMinder documentation. [online]: Zoneminder, 2014 [cit. 2018-03-25]. Dostupné z: http://zoneminder.readthedocs.io/en/stable/userguide/cameracontrol.html

[6] Components – ZoneMinder Documentation [online]: Zoneminder, 2014 [cit. 2018-03-25]. Dostupné z: https://zoneminder.readthedocs.io/en/latest/userguide/components.html

[7] Zdrojový kód zms.cpp. [online]: Zoneminder, 2018 [cit. 2018-04-03]. Dostupné z: https://github.com/ZoneMinder/zoneminder/blob/release-1.29.0/src/zms.cpp

[8] Zdrojový kód zm_monitor.cpp. [online]: Zoneminder, 2018 [cit. 2018-04-03]. Dostupné z: https://github.com/ZoneMinder/zoneminder/blob/release-1.29.0/src/zm_monitor.cpp

[9] Zdrojový kód zm_event.cpp.[online]: Zoneminder, 2018 [cit. 2018-04-04]. Dostupné z: https://github.com/ZoneMinder/zoneminder/blob/release-1.29.0/src/zm_event.cpp

[10] Fórum projektu Zoneminder. [online]: 2018 [cit. 2018-04-03]. Dostupné z: https://forums.zoneminder.com/viewtopic.php?p=16399#p16399

[11] Zdrojový kód zm_stream.h. [online]: Zoneminder, 2018 [cit. 2018-04-04]. Dostupné z: https://github.com/ZoneMinder/zoneminder/blob/release-1.29.0/src/zm_stream.h

[12] ResizeObserver specifikace. [online]: W3C, 2018 [cit. 2018-04-04]. Dostupné z: https://wicg.github.io/ResizeObserver/

[13] MJPEG (Motion JPEG)Video Codec. [online]: The Library of Congress, 2017 [cit. 2018-04-01]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000063.shtml

[14] Cambozola Streaming Image Viewer [online]: Cambozola, 2005 [cit. 2018-04-19]. Dostupné z: http://www.charliemouse.com/code/cambozola/

[15] Mono compatibility. [online]: Mono Project, 2018 [cit. 2018-04-25]. Dostupné z:https://www.mono-project.com/docs/about-mono/compatibility/

[16] PHP na IIS. [online]: Microsoft, 2018 [cit. 2018-04-18]. Dostupné z: https://php.iis.net/

[17] AJAX. [online]: Mozilla, 2018 [cit. 2018-04-21]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

[18] Btrfs Mount Options. [online]: Btrfs, 2017 [cit. 2018-04-01]. Dostupné z: https://btrfs.wiki.kernel.org/index.php/Manpage/btrfs(5)#MOUNT_OPTIONS

[19] The Mono Runtime. [online]: Mono Project, 2018 [cit. 2018-04-18]. Dostupné z:https://www.mono-project.com/docs/advanced/runtime/